Implementation of Hyperspectral Image Classification Algorithms for Brain Tumour Detection using Graphical Processing Units (GPUs)

A. H. Guedes[1], H. Fabelo and G. M. Callicó[1]
[1]Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, Spain

***Abstract*—Graphic Processing Units are suitable platforms to accelerate the classification of hyperspectral images tasks which are an emerging technology for medical diagnosis. Random Forest has proved to be a great candidate in order to classify hyperspectral images. The goal of this paper is focused in the Random Forest training phase acceleration using GPUs, starting from an efficiently CPU implementation of this algorithm. We present multiple bottlenecks identified in the training phase and their solution in order to accelerate them. The different bottleneck solutions achieved in this research study have demonstrated that GPU acceleration is promising in order to generate models in a shorter time, giving the possibility to perform this process in real-time in the not too distant future.**

Keywords - Hyperspectral imaging; Supervised Learning; High Performance Computing; Graphical Processing Units; Random Forest

I. Introduction

Hyperspectral imaging (HSI) refers to the technology that integrates conventional imaging and spectroscopy methods to obtain both spatial and spectral information of an object [1]. HSI sensors measure the radiance of the materials within each pixel area at a very large number of contiguous spectral wavelength bands [2] [3]. The basic task underlying many HSI applications is to identify different materials based on their reflectance spectrum. HSI is an emerging imaging modality for medical applications, especially in disease diagnosis and image-guided surgery [4]. This technology shows some advantages compared to the currently techniques employed for cancer detection, such as Magnetic Resonance (MR), Computed Tomography (CT), Ultrasound (US) and Positron Emission Tomography (PET). The long-term goal of hyperspectral imaging in cancer detection is to develop a simple-to-use, non-invasive, and risk-free tool that will provide early and affordable detection of potentially life threatening malignant tumours. This technology can be use both for screening enhancement and for quantitative analysis of tissue [5]. Since HSI collects high amount of data, it is needed the utilization of high-performance computer platforms where the processing algorithms are implemented. Nowadays, machine learning is used in many research fields because it offers automated procedures that it allows to predict a behaviour based on multiples past observations. The purpose of this work is to use the machine learning for classification of hyperspectral images using Random Forest (RF) [6], implementing this algorithm in a GPU so as to accelerate the critical parts of the training phase to process in-vivo human brain hyperspectral images. This work is framed in the European Project HELICoiD “HypErspectral Imaging Cancer Detection” (FP7-618080) [7].

II. Materials and Methods

A. *In-vivo human brain tumour database*

This study uses the hyperspectral images obtained by the acquisition system developed within the HELICoiD project at the University Hospital Doctor Negrín of Las Palmas de Gran Canaria. The hyperspectral acquisition system consists of two hyperspectral cameras coupled together able to capture two different hyperspectral cubes in the VNIR (Visible and Near Infrared) and NIR (Near Infrared) spectral range. This work is focused in the images obtained in the VNIR range. The dataset is composed by 6 different images from 4 different patients affected by a glioblastoma (GBM) tumour. The samples have been pre-processed reducing the number of bands (features) to 129 and have been labelled in 4 different classes (Normal Tissue, Tumour Tissue, Blood Vessels and Background) obtaining a total of 87,722 in-vivo human brain samples.

B. *Random Forest for Tumour Detection*

The RF CPU implementation has been based on an existent implementation called Ranger (RANdom Forest GeneRator) [8]. The core of Ranger is implemented in C++ and uses only standard libraries. Ranger has been verified using the previously described database. K-Fold Cross-Validation method has been employed for the evaluation with 10 folds. Figure 1 presents the average of the evaluation metrics (sensitivity, specificity and overall accuracy) for the different classes performed to the dataset. These results outperform 99% of overall accuracy for every class providing good discrimination between the different classes.

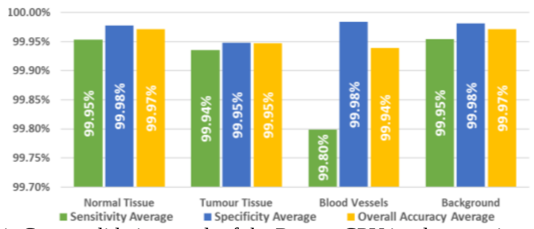

Figure 1: Cross-validation result of the Ranger CPU implementation with a forest of 500 trees

C. *Random Forest Parallelization Analysis*

Ranger has a good performance analysis in the different bottleneck of RF algorithm. However, we identified multiple bottleneck of Ranger that could be accelerated using GPUs. The first bottleneck will accelerate the forest initialization. We proposed to generate the learning set of different trees during the initialization of the forest in order to construct multiple trees by setting the learning set. It can be only used if Ranger option generates a bootstrap learning set without replacement. The principal bottleneck of Ranger is within the splitting process, during the grow phase of a tree, to compute node impurity. Non-terminal nodes need to splitting and is necessary to identify which feature is better, between all possible candidate, and what value to use for threshold. the problem of locating the best split scale proportionally to the number of samples and the number of possible features.

D. *GPU Implementation*

GPU kernels have been implemented in order to get a better performance in the RF algorithm. In the *bootstrap kernel*, the idea is that each block generates the training dataset for each tree. The solution is a one dimensional grid where each block generates the learning set of a tree (Figure 2). As training set is generated randomly from dataset, cuRAND library has been used. The most critical part that we identified is in the method which generates a new split node, *findBestSplit* method. This process is divided in three kernels: first kernel generates an overall class count from the dataset, second kernel generates an overall class count of the possible right child nodes and the last kernel computes the decrease of impurity of the node. The second kernel uses a two-dimensional grid where the number of blocks in X axis is the number of possible features and Y axis is fixed (Figure 3). The other kernels use a one-dimensional grid with a fixed number of blocks in X axis.

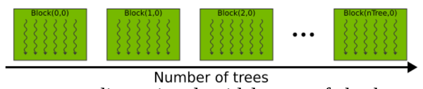

Figure 2: Bootstrap one-dimensional grid layout of the bootstrap kernel for *nTree* number of trees.

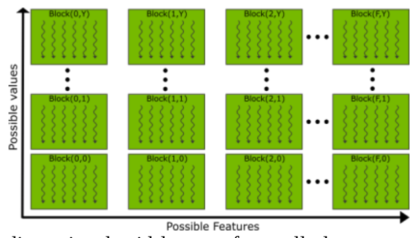

Figure 3: Two-dimensional grid layout of overall class count in the possible right node child. We use the X coordinate for the feature response and the Y coordinate assign a threshold.

III. Experimental Results

This section presents the results achieved in the different bottleneck solutions. Two different equipment have been used for these tests, a laptop (Intel Processor i7-6700HQ and a NVIDIA GPU GTX 960M) and a server (Intel Xeon Processor E3-1225 v3 and a NVIDIA GPU Tesla K40). Figure 4 shows the comparison between the results generated using both platforms. Figure 5 shows the graphical comparison between the results obtained using the laptop versus the server using multiple *possible features* (3, 6, 12, 18 and 24) in the *findBestSplit kernel.*

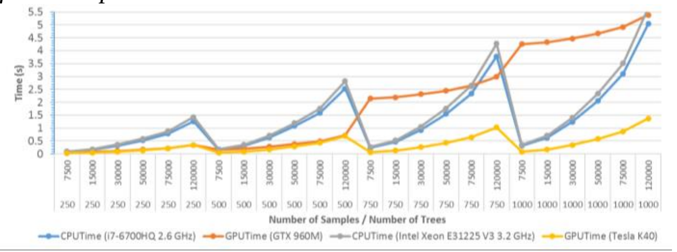

Figure 4: Testing *bootstrap kernel* comparison (GTX 960M vs. Tesla K40).

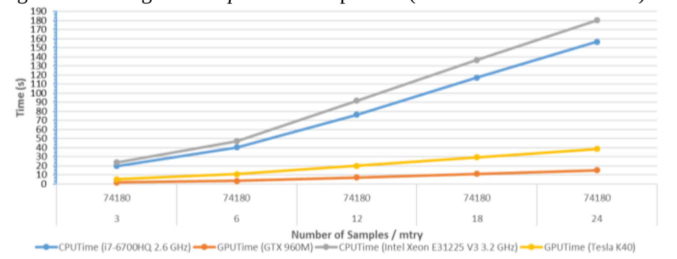

Figure 5: Testing *findBestSplit kernel* comparison with 74,180 samples in the non-terminal node (laptop vs. IUMA's server).

IV. Conclusions

This research work presents a comparison between two kernels developed to accelerate the training process in the RF algorithm using two different platforms employing in-vivo human brain hyperspectral samples. The *bootstrap kernel* offers better results in Tesla K40 than GTX 960M when the number of trees is higher than 750. This result could depend of the number of Stream Multiprocessor (SMX) available and the synchronization time of the different blocks. However, *findBestSplit kernel* has better performance in GTX 960M than Tesla K40, due to this kernel has a grid with less variability than the Tesla kernel, the number of blocks in Y axis is fixed and the range of blocks in X axis is much smaller than the range in the bootstrap kernel. Although the kernel is misusing resources of Tesla K40, we have demonstrated that the *findBestSplit kernel* works correctly and it has a better performance than CPU comparison.

References

[1] Li, Q., He, X., Wang, Y., Liu, H., Xu, D., & Guo, F. (2013). Review of spectral imaging technology in biomedical engineering: achievements and challenges. Journal of biomedical optics, 18(10), 100901-100901.

[2] Manolakis, D., Marden, D., & Shaw, G. A. (2003). Hyperspectral image processing for automatic target detection applications. Lincoln Laboratory Journal, 14(1), 79-116.

[3] Manolakis, D., & Shaw, G. (2002). Detection algorithms for hyperspectral imaging applications. Signal Processing Magazine, IEEE, 19(1), 29-43.

[4] Lu, G., & Fei, B. (2014). Medical hyperspectral imaging: a review. Journal of biomedical optics, 19(1), 010901-010901.

[5] A. Sahu et al., "Characterization of Mammary Tumors Using Noninvasive Tactile and Hyperspectral Sensors," in IEEE Sensors Journal, vol. 14, no. 10, pp. 3337-3344, Oct. 2014.

[6] Gilles Louppe, "Understanding Random Forests", Jul. 2014

[7] H. Fabelo, S. Ortega, R. Guerra, G. Callicó, A. Szolna, J. F. Piñeiro, M. Tejedor, S. López, R. Sarmiento. A Novel Use of Hyperspectral Images for Human Brain Cancer Detection using in-Vivo Samples. In Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies. 311-320 (2016).

[8] Marvin N. Wright and Andreas Ziegler, "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R", Aug. 2015.